Lab 2

Reg. No: 19BCE1209

Name: Gautam Sanjay Wadhwani

Course: CSE4001 Parallel and Distributed Computing

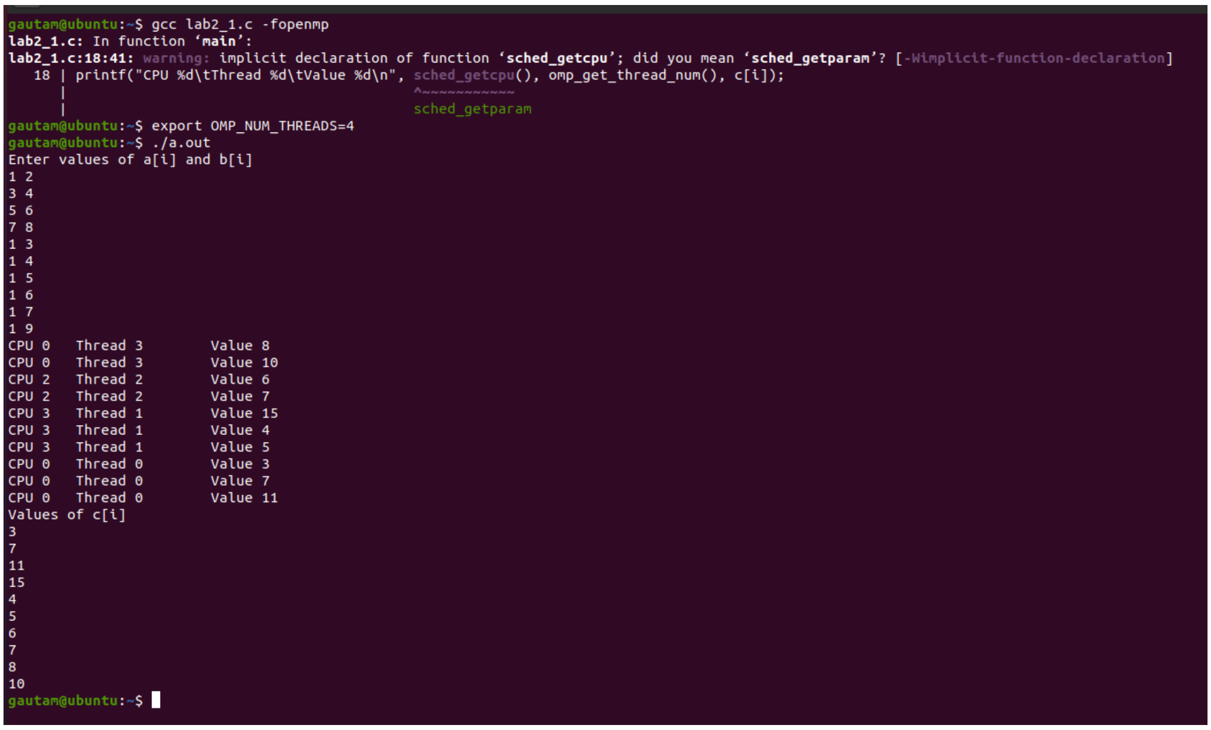
Q1. Array Addition using Parallel For

Code:

```
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <omp.h>
#include<sched.h>

int main() {
int a[10], b[10], c[10];
int i;
printf("Enter values of a[i] and b[i]\n");
for(i = 0; i < 10; i++) {
scanf("%d %d", &a[i], &b[i]);
}
#pragma omp parallel for
for (i = 0; i < 10; i++)
{
c[i] = a[i] + b[i];
printf("CPU %d\tThread %d\tValue %d\n", sched_getcpu(), omp_get_thread_num(), c[i]);
}
printf("Values of c[i]\n");
for(i = 0; i < 10; i++) {
printf("%d\n", c[i]);
}
return 0;
```

}

Output:



```
gautam@ubuntu:~$ gcc lab2_1.c -fopenmp
lab2_1.c: In function ‘main’:
lab2_1.c:18:41: warning: implicit declaration of function ‘sched_getcpu’; did you mean ‘sched_getparam’? [-Wimplicit-function-declaration]
   18 | printf("CPU %d\tThread %d\tValue %d\n", sched_getcpu(), omp_get_thread_num(), c[i]);
      |                                         ^~~~~~~~~~~~
      |                                         sched_getparam
gautam@ubuntu:~$ export OMP_NUM_THREADS=4
gautam@ubuntu:~$ ./a.out
Enter values of a[i] and b[i]
1 2
3 4
5 6
7 8
1 3
1 4
1 5
1 6
1 7
1 9
CPU 0   Thread 3        Value 8
CPU 0   Thread 3        Value 10
CPU 2   Thread 2        Value 6
CPU 2   Thread 2        Value 7
CPU 3   Thread 1        Value 15
CPU 3   Thread 1        Value 4
CPU 3   Thread 1        Value 5
CPU 0   Thread 0        Value 3
CPU 0   Thread 0        Value 7
CPU 0   Thread 0        Value 11
Values of c[i]
3
7
11
15
4
5
6
7
8
10
gautam@ubuntu:~$
```

Q2. Sample for Private Variable

Code:

```
#include <stdio.h>

#include <pthread.h>

#include <stdlib.h>

#include <omp.h>

int main() {

int numThreads, tid;

#pragma omp parallel private(tid)

{

tid = omp_get_thread_num();

printf("Hello World from thread = %d\n", tid);

if(tid == 0) {
```

numThreads = omp_get_num_threads();

printf("Number of threads = %d\n", numThreads);

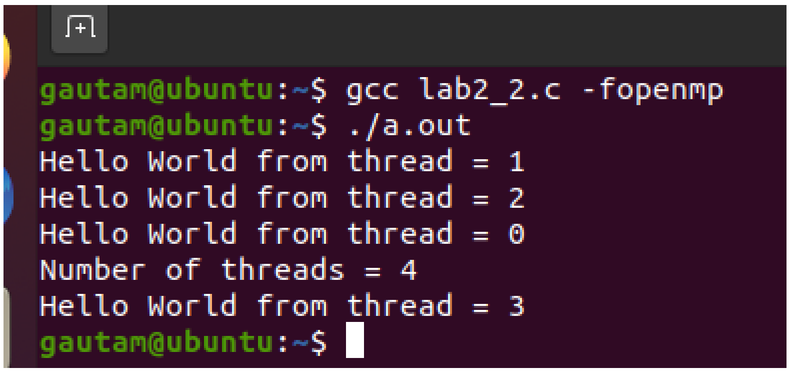
}

}

return 0;

}

Output:



```
gautam@ubuntu:~$ gcc lab2_2.c -fopenmp
gautam@ubuntu:~$ ./a.out
Hello World from thread = 1
Hello World from thread = 2
Hello World from thread = 0
Number of threads = 4
Hello World from thread = 3
gautam@ubuntu:~$
```

Q3. Array addition using parallel for with a,b, c as private arrays

Code:

#include <stdio.h>

#include <pthread.h>

#include <stdlib.h>

#include <omp.h>

#include<sched.h>

int main() {

int a[10], b[10], c[10];

int i;

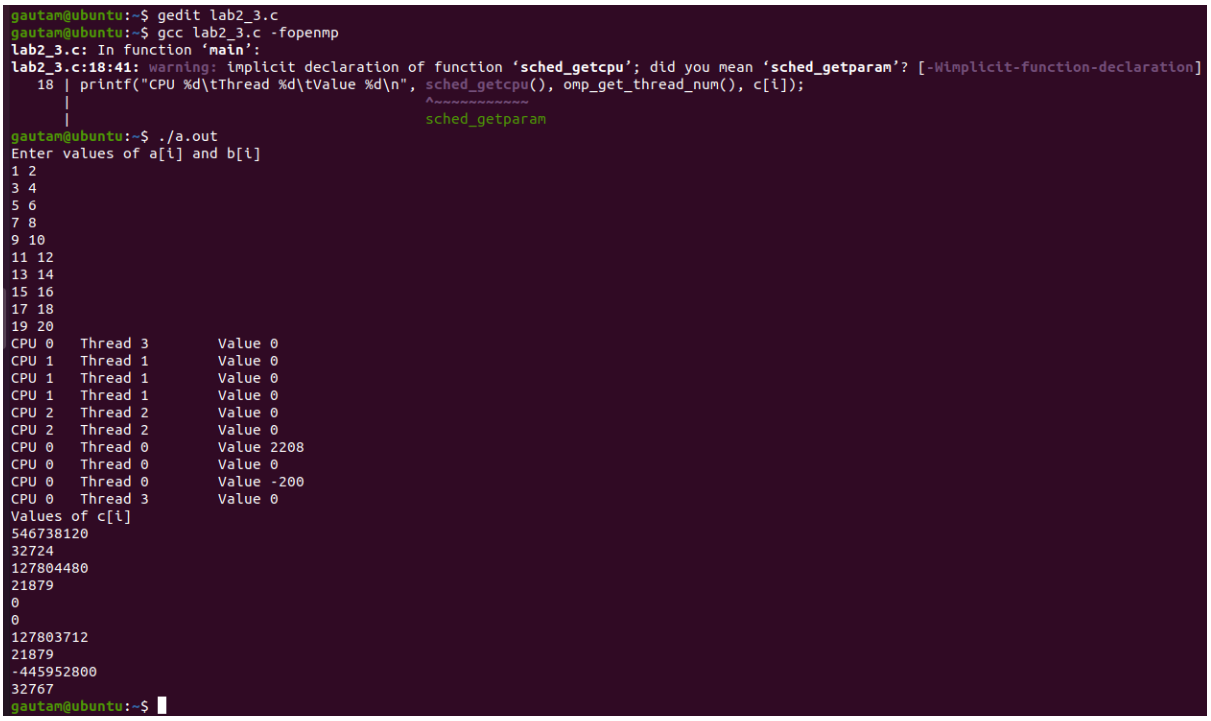
printf("Enter values of a[i] and b[i]\n");

for(i = 0; i < 10; i++) {

scanf("%d %d", &a[i], &b[i]);

```
}
#pragma omp parallel for private(a,b,c)
for (i = 0; i < 10; i++)
{
c[i] = a[i] + b[i];
printf("CPU %d\tThread %d\tValue %d\n", sched_getcpu(), omp_get_thread_num(), c[i]);
}
printf("Values of c[i]\n");
for(i = 0; i < 10; i++) {
printf("%d\n", c[i]);
}
return 0;
}
```

Output:



```
gautam@ubuntu:~$ gedit lab2_3.c
gautam@ubuntu:~$ gcc lab2_3.c -fopenmp
lab2_3.c: In function ‘main’:
lab2_3.c:18:41: warning: implicit declaration of function ‘sched_getcpu’; did you mean ‘sched_getparam’? [-Wimplicit-function-declaration]
   18 | printf("CPU %d\tThread %d\tValue %d\n", sched_getcpu(), omp_get_thread_num(), c[i]);
      |                                         ^~~~~~~~~~~~
      |                                         sched_getparam
gautam@ubuntu:~$ ./a.out
Enter values of a[i] and b[i]
1 2
3 4
5 6
7 8
9 10
11 12
13 14
15 16
17 18
19 20
CPU 0   Thread 3        Value 0
CPU 1   Thread 1        Value 0
CPU 1   Thread 1        Value 0
CPU 1   Thread 1        Value 0
CPU 2   Thread 2        Value 0
CPU 2   Thread 2        Value 0
CPU 0   Thread 0        Value 2208
CPU 0   Thread 0        Value 0
CPU 0   Thread 0        Value -200
CPU 0   Thread 3        Value 0
Values of c[i]
546738120
32724
127804480
21879
0
0
127803712
21879
-445952800
32767
gautam@ubuntu:~$
```

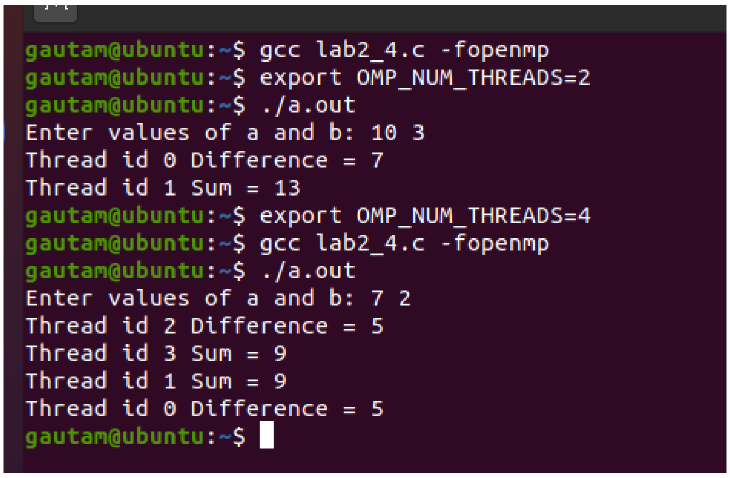
This does not work because, different threads have their own copy of c which they update, and hence, the copy of c with the main thread is not updated

Q4. Parallelize addition and subtraction of two integer variables a and b

Code:

```
#include <stdio.h>

#include <pthread.h>

#include <stdlib.h>

#include <omp.h>


int main() {

  int a, b, sum, difference, id;

  printf("Enter values of a and b: ");

  scanf("%d %d", &a, &b);

  #pragma omp parallel shared(a,b) private(sum, difference, id)

  {

     id = omp_get_thread_num();

     if(id%2) {

     sum = a+b;

     printf("Thread id %d Sum = %d\n", id, sum);

     }

     else {

     difference = a-b;

     printf("Thread id %d Difference = %d\n", id, difference);

     }

  }

}
```

Output:



```
gautam@ubuntu:~$ gcc lab2_4.c -fopenmp
gautam@ubuntu:~$ export OMP_NUM_THREADS=2
gautam@ubuntu:~$ ./a.out
Enter values of a and b: 10 3
Thread id 0 Difference = 7
Thread id 1 Sum = 13
gautam@ubuntu:~$ export OMP_NUM_THREADS=4
gautam@ubuntu:~$ gcc lab2_4.c -fopenmp
gautam@ubuntu:~$ ./a.out
Enter values of a and b: 7 2
Thread id 2 Difference = 5
Thread id 3 Sum = 9
Thread id 1 Sum = 9
Thread id 0 Difference = 5
gautam@ubuntu:~$
```

This program calculates sum for odd thread id and difference for even thread id